TEST CASE SELECTION FOR PATH TESTING USING BEE COLONY OPTIMIZATION

M.Mounika¹, D.Vivekananda Reddy²

¹M.Tech, Department of Computer Science, Sri Venkateswara University, Tirupati, Andhra Pradesh, India.
²Assistant Professor, Department of Computer Science, Sri Venkateswara University, Tirupati, Andhra Pradesh, India.

Abstract- In software development life cycle (SDLC), testing phase is the most important phase. Without testing we can’t give quality software or risk free software to the client. Software testing process typically consumes at least 50% of the total cost involved in software development. In regression testing there evolves the number of test cases. Due to some constraints, it is impractical to test all of them. Therefore to overcome this problem, testing is done using selected test cases to reduce the testing effort and get the desired result accurately. We present the selection of test cases with the help of Bee Colony Optimization (BCO) Algorithm. BCO is a population-based search algorithm inspired by the natural foraging behaviour of honey bees to find the optimal solution. The Algorithm proposed here uses swarm based intelligence technique on Maximum path coverage to select the test cases. Independent paths are generated with the help of Control flow graphs which are useful in path testing. We prioritize or select the test cases based on criteria or quality or fitness value of path generated by CFG. This approach guarantees full path coverage. In this approach, we extend the functionality of the bee to do testing and monitoring activity so that it reduces the manual work and improves the confidence on the software by testing it with coverage of the given software. In this approach we used path coverage and condition coverage as coverage based test adequacy criteria.

1. INTRODUCTION

Testing is the process of evaluating a system or its component to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements. Testing is a process of verification and validation of the software product for its correctness and accuracy of working [18, 1].

In Software Development Life Cycle (SDLC), Testing Phase is the important phase to develop the high quality software product. Software Testing is labour intensive process which also associated with high cost. Exhaustive testing is not possible. So testers choose subsets of tests that can uncover the maximum number of errors. Testing can only prove the presence of defect, never their absence.

As software once delivered comes in maintenance phase where it undergoes constant change and to test these changes we use Regression testing.

Regression Testing is done to ensure that enhancements or defect fixes made to the software works properly and does not affect the existing functionality. Regression Testing enables that any new feature introduced to the existing product does not adversely affect the current functionality. Whenever the defect fixes are done, a set of test cases that need to be run to verify the defect fixes are selected by the test team. An impact analysis is done to find out what areas may get impacted due to those defect fixes. Based on the impact analysis, some test cases are selected to take care of the impacted areas. Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated tools. This is generally performed in maintenance phase of SDLC [1].

Due to the resource and time constraints for re-executing large test suites, it is mandatory to optimize available test suites by using test cases prioritization, test case filtration, test case selection and test suite minimization. This paper is focused on path testing, a form of functional testing which concerns about maximum coverage of path. In regression testing there is large number of test cases, but due to time and cost constraint it’s not possible to execute all of them. So test case optimization is done to reduce the number of test cases and select the effective test cases.

Path testing ensures that every possible logical execution path in a program must be exercised at least once. The path coverage strategy requires us to design test cases such that all independent paths in the program are executed at least once. Independent path can be defined in terms of the control flow graph (CFG) of a program.

A control flow graph describes the sequence in which the different instructions of a program get executed. In other words, a control flow graph describes how the control flows through the program. In order to draw the control flow graph of a program, all the statements of a program must be numbered first. The different numbered statements serve as nodes of the control flow graph. An edge from one node to another node exists if the execution of the statement representing the first node can result in the transfer of control to the other node. Each independent path would comprise number of normal nodes and predicate nodes [21].

Independent paths are very useful in testing the path coverage. Paths generated by CFG are traversed by test cases to be prioritized based on some test adequacy criteria or fitness value. The meta-heuristic methods tried in past to solve this problem yielded success, but with serious drawbacks. The BCO which is swarm based meta-heuristic algorithm for optimizing problems. BCO has been used for solving large and complex real-world problems. It also finds application in the
field of software testing, which is one of the most important phase in software development lifecycle. BCO is used to generate the optimal number of test-cases which are sufficient to cover the paths generated by using the CFG (control flow graph). The independent paths are considered to be an important criterion to generate the effective test cases for path coverage. The effective test-cases are those which cover all the nodes of the path in less time.

2. RELATED WORK

The application of artificial intelligence (AI) techniques in Software Engineering (SE) is an emerging area of research that brings about the cross fertilization of ideas across two domains. The research on software testing problems has concentrate mostly on software test optimization. So intelligent software testing techniques are extensively proposed to solve software test optimization problems. Several meta-heuristic search techniques such as applying rules, meta-heuristics (like Genetic Algorithm (GA), Ant Colony Optimization (ACO), Tabu Search, Simulated Annealing, Bacteriologic Algorithm (BA), etc.), Fuzzy logic and Neural Networks (NN), Hybrid Genetic Algorithm (HGA) and other approximation methods have been proposed for some specific types of problems in test optimization.

Genetic algorithm, a adaptive search procedure is introduced by John Holland, broadly studied by Goldberg and De Jong. It is an optimization technique which provides near optimal solution. Genetic Algorithm is based on the idea on the natural evolution. The foundation of GA lies on the concept of the survival of fittest into a solution space. Each cycle of GA process includes initialization (encoding), selection based on fitness function, reproduction using crossover or mutation. The cycle is repeated till a solution is found that satisfies the minimum criteria or a fixed number of generations has been reached [14].

But there are some disadvantages of GA that it’s not so effective to solve a problem involving single measure for evaluation, external optimization, not sufficient to converge to a solution which cause results are to consistent.

Ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Marco Dorigo[20], proposed ant colony Optimization in 2005. ACO has its real strength in the overwhelming behavior of ants looking for a path between their colony and a source of food. Ants are blind and they communicate with their colony by a chemical substance called pheromone. Ants while moving yields pheromone along the path traversed. This pheromone trail is sensed by other ants through which other ants follow the path with maximum pheromone trail. The process is repeated till sufficient amount of food has been gathered. This substance evaporates with time with some amount of percentage [16].

Demerit of ACO is it waste large amount of computing resource to generate test cases. In ACO converges is guaranteed but time to convergence are uncertain, length of test sequence is higher and repetition of nodes in same sequence.

There is still a need for a more effective solution approach to more general problems such as test suite optimization and there is a need for developing models and efficient algorithms to achieve optimization by satisfying the specified test adequacy criteria.

The proposed algorithm is used to generate optimal number of test cases and to achieve maximum coverage of paths. Here we used BCO (Bee Colony Optimization) Algorithm where bees finding the food sites (test cases) then calculate their quality (fitness) which used to identify the best test cases with maximum coverage and less computing resources.

3. BEE COLONY OPTIMIZATION: BCO-APPROACH

BCO is the name given to the colony formed by the mutual understanding of the natural bees in the food for aging process. All creatures on this earth follow one or the other mechanisms/process to find food source that suite them and these mechanism are found in insects like ants, bees. Honey bee comb build-up and management is a classic example of team work, experience, coordination and synchronization. These are the factors which have given rise to interest of researches to find solutions to their problems [12].

Dervis Karaboga in 2005 defined BCO, which is swarm based meta-heuristic algorithm for optimizing problems. BCO has been used for solving large and complex real-world problems. BCO poses an ability to find high quality solutions of difficult combinatorial problems within a reasonable amount of computer time. The BCO is a stochastic, random-search technique. This technique uses an analogy between the way in which bees in nature search for a food, and the way in which optimization algorithms search for an optimum of given combinatorial optimization problems. The BCO works in a self-organized and decentralized way and therefore represents a good basis for parallelization [11].

The bee colony algorithm consists of 3 types of bees-
- employee bee / Search bee
- onlooker bee / Selector bee
- scout bee / Replace bee

Scout bee is responsible for carrying out random searches in the environment. A bee who visits the food source visited by it previously is called an employed bee and the bee that waits in the hive for decision making is called the onlooker bee.

In the BCO algorithm, each cycle of the search consists of three steps: In the beginning, some food sources are randomly selected by the bees and the amount of nectar is also determined. Then these bees return to the hive and share this information by performing the waggle dance. In second stage, each employed bee goes to the food source visited by her in previous cycle then by means of visual information a new food source in the neighborhood is determined. In the third stage, an
onlooker bee visits the food source position depending on the nectar information shared by the employed bees. The food source with maximum nectar quantity is selected by the onlooker bee. After arriving at the selected food source, the onlooker bee according to visual information chooses a new food source in the neighborhood of the selected food source. If the fitness value of the new food source is higher than that of the previous one, the bee memorizes the new food source and removes the old one. Otherwise it keeps the position of the old food source in the knowledge base. Once the food source is abandoned by the bee, a new food source is randomly selected by a scout and then the abandoned source is replaced by this new food source by scout bees [7].

The Main steps of BCO Algorithms are:

1. Initial food sources are produced for all employed bees.
2. Each employed bee goes to a food source in her memory and determines a neighbor source, then evaluates its nectar amount and dances in the hive.
3. Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source. After choosing a neighbor around that, she evaluates its nectar amount.
4. Abandoned food sources are determined and then, they are replaced with the new food sources discovered by scouts.
5. The best food source found so far is registered. UNTIL (requirements are met)

![Flow diagram of BCO](image)

Figure 1: Flow diagram of BCO

BCO has been used for solving large and complex real-world problems they are

- Job Scheduling
- Robotics
- Travelling Salesman Problem
- Data Mining
- Statistical Quality Control
- Wood Defect Classifier
- Mechanical Design
- Electronic Design
- Clustering etc.

4. APPLICATION OF BCO IN SOFTWARE TESTING

Software testing is a type of multi variable optimization problem where generation and selection of efficient test cases cannot be achieved within permissible time bounds. Hence for solving these types of problems, meta-heuristics search algorithms have been proposed. These algorithms help in finding the near optimal solution in reasonable running time. The artificial bee colony algorithm, which is also a meta-heuristics search algorithm, is capable of locating efficient solutions. The algorithm models the food foraging behavior of honey bees. The main focus of software testing is on uncovering as many errors as possible in the given time, as this would help in conforming the product to the requirement specifications and also to validate the quality of the software produced.

Initially the program converts into a corresponding Control Flow Graph (CFG). Now from control flow graph, the independent paths from the start node to the end node are generated. Each independent path would comprise number of normal nodes and predicate nodes. Every independent path would represent a Test Case. Now the BCO algorithm is applied to generate an Optimal Test suite by generating optimal test data which would traverse through the independent paths and hence into to the test cases [9].

Initialize the Test Cases by initial test data generated by the equivalence partitioning and boundary value analysis methods.

**Step 1:** The Search bee employ test cases to the first executable node of the program. The fitness value of the test cases and the node information is returned to the employed bees.

**Step 2:** The Selector Bee takes the node information and fitness value of the test cases as the input and evaluates the fitness of each test case taken from the employed bees. Then the test case with the highest fitness value is stored in the memory or repository.

**Step 3:** Then the neighbouring nodes of the covered nodes are searched by the employed bees and the fitness value of the current test case is explored against the neighbouring nodes by the onlooker bee.

**Step 4:** The node with the highest fitness value is picked out and tagged with the current node which forms the test path.
This information of the test path is stored in the Optimal Test repository.

**Step 5:** Rest of the nodes other than the covered node and the test cases other than the selected test case are abandoned and they are stored in a temporary abandoned repository.

**Step 6:**
Repeat the steps 3 to 6 till the test path is not complete

Else the nodes and the test cases from the abandoned repository are selected for the next test path generation by the employed bees

**Step 7:** The scout bee generate a new population of test cases and replaces the test cases of the abandoned repository with the new test cases if the onlooker bee finds the selected test cases are not efficient.

Until the user defined termination criterion is met.

### 5. PSEUDO CODE FOR BCO ALGORITHM

1. Initialize the population of test cases $x_{ij}$.
2. Evaluate the population
3. Cycle = 1
4. Repeat
5. Produce new test cases $v_{ij}$ in the neighborhood of $x_{ij}$ for the employed bees using the formula
   
   $$v_{ij} = x_{ij} + q_{ij} (x_{ij} - x_{kj})$$

   Where $k$ is a solution in the neighborhood of $i$, $q_{ij}$ is a random number in the range [-1, 1] and evaluate them.
6. Apply greedy selection process between $x_i$ and $v_i$.
7. Calculate the probability values of test cases $x_i$ by means of their fitness values using the equation:
   
   $$P_i = \frac{fit_i}{\sum fit_i}$$

   Where $i=1$ to SN
8. Produce new test cases $v_i$ for the onlookers from the test cases $x_i$, selected depending on $P_i$ and evaluate them.
9. Apply the greedy selection process for the onlookers between $x_i$ and $v_i$.
10. Determine the abandoned test case, if exists and replaces it with a new randomly produced test case $x_i$ for the scout using the equation:

   $$x_{ij} = \min_j + \text{rand}(0,1) (\max_j - \min_j)$$

   The scout waggle dances to indicate the new test case generation.
11. Memorize the best test case achieved so far using the fitness value.
12. Cycle = cycle+1
13. Until cycle=Max. Cycle Number (MCN)

### 6. CASE STUDY

Sample problem taken is Quadratic Equation solving program. Here the decision making process is done based on the value of the parameters a, b and c. The data types of all these variables are integer.

Sample program (SUT) is

1. ```c
   int main()
   {
   int A,B,C;
   float disc,deno,x1,x2;
   printf("PROGRAM TO FIND THE ROOTS OF A QUADRATIC EQUATION");
   printf("ENTER THE VALUES OF A,B,C...");
   scanf("%d, %d, %d", &A, &B,&C);
   disc = (B*B)-(4*A*C);
   deno = 2*A;
   if(disc > 0)
   
   Table 1 Independent paths and initial test cases
   ```

---

**Fig 2 Working principle of BCO**
Initial set of test cases for each of these paths are randomly created.
1. Initial population of test cases \( x_i \) for each independent paths. Let \( x_i \) be the initial set of test cases for each of the path as below:
\[ x_i = \{1,4,3\} \text{ for path 1, } \{2,4,3\} \text{ for path 2, } \{2,3,4\} \text{ for path 3} \]
2. Evaluate the population
3. Cycle=1
4. Repeat
5. Produce new test cases \( v_{ij} \) neighborhood of \( x_{ij} \)
\[ V_{ij} = x_{ij} + q_{ij}(x_{ij} - x_{kj}) \]
For Path 1:
- \( i=1 \) to 3, \( j=1 \), \( k = (i+1) \% \text{ size of the test case} \) (size = 3 since we have 3 variables)
- \( v_{11} = x_{11} + q_{11}(x_{11} - x_{21}) = 1 + (0)(1 - 4) = 1 \)
- \( v_{21} = x_{21} + q_{21}(x_{21} - x_{31}) = 4 + (0)(4 - 3) = 4 \)
- \( v_{31} = x_{31} + q_{31}(x_{31} - x_{11}) = 3 + (1)(3 - 1) = 5 \)
Hence, \( vi = \{1,4,5\} \) where \( i=1 \) and \( j=1 \)
6. Compare the fitness of \( x_{ij} \) and \( v_{ij} \)
7. Calculate the probability for test cases
\[ P_i = \frac{\text{fit}_i}{\text{sum of fit}} \]
8. In this example test case \( \{1,4,3\} \) has 100% fitness value where \( \{1,4,5\} \) has 0% fitness value. test cases with least fitness value are abandon.
9. Determine the abandon solution and replace it with randomly produced solution \( x_i \) for the scout using the formula
\[ X_i = \text{min}_j + \text{rand}(0,1)*\text{(max}_j - \text{min}_j) \]
10. Go to step 4 and repeated the process for other independent paths.
Where fitness values is 100 for complete traversal and 0 for incomplete. So, the optimal test suite is generated successfully for the above program.

7. COMPARISION

Using the swarm based intelligence algorithm there are exist a probability of falling into a Local optimum but using Bee colony optimization algorithm probability is low. As BCO work as combination of both local and global search ability with main aim is to improve. Here is some figure which shows that using BCO approach it achieves maximum coverage based on Number of Cycles, percentages of path covered which increases and maintain consistency after a fix threshold value.

<table>
<thead>
<tr>
<th>S.No</th>
<th>path</th>
<th>test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-2-6</td>
<td>(1,4,3)</td>
</tr>
<tr>
<td>2</td>
<td>1-3-5-6</td>
<td>(2,4,3)</td>
</tr>
<tr>
<td>3</td>
<td>1-3-4-6</td>
<td>(2,3,4)</td>
</tr>
</tbody>
</table>

The SUT uses 3 inputs of a quadratic equation and generates its roots.

The Control Flow Graph (CFG) for the SUT is given below.

![Figure 2: Flow graph of the quadratic equation](image)

From CFG different independent paths are generated. Three independent paths are generated i.e 1-2-6,1-3-4-6,1-3-5-6.
The graph from Figure 4 plots the number of runs and path showing the paths which increased incrementally. The line of graph is having lesser slope. The Figure 5, graph plots comparison between BCO, ACO and GA [2] between number of non-feasible paths and the LOC. This graph testifies that the BCO distinguishes and accounts less number of non-feasible paths. Thus more number of feasible paths with test data is covered. But GA doesn’t find any non-feasible paths. Non-feasible paths are the paths which cannot be processed using any generated test data. Hence, the non-feasible path is technically accessible but logically cannot be processed. This approach of BCO will return non feasible paths if any path generated using BCO are not accessible using any test data.

8. SCREEN SHOTS

9. CONCLUSION

In this paper, independent paths are generated and the test case selection is done based upon Bee Colony Optimization (BCO). We investigated the performance of standard of the Bee Colony algorithm and compared their performances against other swarm based intelligence algorithm. Also the results’ showing that BCO approach is much better than the other optimizing algorithm. BCO approach is more efficient and work faster for optimizing test cases. The proposed BCO algorithm has been explained using example on generating independent paths and maximum coverage or maximum path coverage. The employee bee looks for path until they get 100%
coverage test data. In future, there different other factors optimized test case results

REFERENCES


